Short Paper: A New Way to Achieve Round-Efficient Asynchronous Byzantine Agreement

Simon Holmgaard Kamp

CISPA Helmholtz Center for Information Security
simon.kamp@cispa.de

Abstract. We translate the *expand-and-extract* framework by Fitzi, Liu-Zhang, and Loss (PODC 21) to the asynchronous setting. While they use it to obtain a synchronous BA with $2^{-\lambda}$ error probability in $\lambda+1$ rounds, we make it work in asynchrony in $\lambda + 3$ rounds. At the heart of their solution is a *proxcensus* primitive, which is used to reach graded agreement with $2^r + 1$ grades in r rounds by reducing proxcensus with $2s - 1$ grades to proxcensus with s grades in one round. The *expand-and-extract* paradigm uses proxcensus to *expand* binary inputs to $2^\lambda + 1$ grades in λ rounds before *extracting* a binary output by partitioning the grades using a λ bit common coin. However, the proxcensus protocol by Fitzi et al. does not translate to the asynchronous setting without lowering the corruption threshold or using more rounds in each recursive step.
We solve this by attaching *justifiers* to all messages: forcing the adversary to choose between being ignored by the honest parties, or sending messages with certain validity properties. Using these we define validated proxcensus and show that it can be instantiated in asynchrony with the same recursive structure and round complexity as synchronous proxcensus. In asynchrony the extraction phase incurs a security loss of one bit which is recovered by expanding to twice as many grades using an extra round of communication. This results in a $\lambda + 2$ round and a $\lambda + 3$ round BA, both with $2^{-\lambda}$ error probability and communication complexity matching Fitzi et al.

1 Introduction

Following the *expand-and-extract* paradigm by Fitzi, Liu-Zhang, and Loss [9] we present a concretely round efficient asynchronous Monte Carlo style BA which runs for a fixed number of rounds, $\lambda+3$, to reach agreement on a binary decision with probability at least $1 - 2^\lambda$ using signatures and a common coin.

The expand-and-extract paradigm generalizes Feldman and Micali [8] (FM) in which the parties iteratively run *crusader agreement*, which outputs either the input of an honest party or an inconclusive value "?" with the guarantee that no two honest parties get different bits as output. Then they flip a coin and parties that had output "?" use the coin as input to the next FM iteration. With probability $> 1/2$ this makes the system enter a *univalent* configuration, in which

honest parties input the same bit to all future iterations. While it is possible to detect this univalent state and terminate in an expected constant number of rounds; in order to enter this state and reach agreement with probability $1 - 2^{-\lambda}$, the protocol must run λ FM iterations which corresponds to 2λ rounds of communication.

Fitzi et al. cut this worst case round complexity almost in half to $\lambda + 1$ by introducing proxcensus which generalizes crusader agreement and graded agreement to any number of grades, while still requiring that all honest outputs are distributed between two adjacent grades. They observe that you can remove all but one of the coin flip rounds by first using proxcensus to *expand* to $2^\lambda + 1$ grades in λ rounds and then flip a single λ-bit coin to *extract* a bit decision based on the value of the coin and each party's grade in a manner that is consistent with probability $1 - 2^{-\lambda}$. Their expansion technique is inherently synchronous, but we adapt it to the asynchronous setting while matching the asymptotic communication complexity and the concrete round complexity up to needing one extra round to implement validated BA (VBA) as defined in [3] or two extra rounds to implement a full-fledged BA.

To compare with the FM approach in the asynchronous setting: observe that crusader agreement followed by a coin flip does not on its own solve BA in asynchrony. This is because the adversary is assumed to learn the output of the coin as soon as some honest party wishes to flip the coin. From a *bivalent* configuration (where at least two honest parties have different inputs) the adversary can make an honest party output "?" in order to learn the coin c and then let other honest parties receive output $1 - c$ to maintain the bivalent configuration indefinitely. This can be solved by using graded agreement with at least 4 grades, which requires an extra round of communication before flipping the coin. The resulting FM iterations use 3 rounds each and give a worst case round complexity of 3λ. The expand and extract technique improves on the worst case round complexity of this asynchronous BA by almost a factor 3 to $\lambda + 3$.

In the synchronous BA by Fitzi et al., the λ rounds spent on flipping a 1-bit coin in each FM iteration are removed and then replaced with a single round used to flip a λ-bit coin after expanding. The resulting round complexity is $\lambda + 1$ instead of 2λ. In the same vein we can view our protocol as removing – from each of the λ FM iterations – both the round spent on coin flipping and the round spent on upgrading crusader agreement to graded consensus with more grades. In the end a (worst case) total of 2λ rounds are replaced by just two rounds after the expansion phase, that each functionally stand in for λ rounds and in combination result in a round complexity of $\lambda + 2$ rather than 3λ.

Techniques. The expand-and-extract approach first *expands* to $2^\lambda + 1$ grades and then uses a λ-bit coin to *extract* a decision. In the expansion step the parties input their bit to a *proxcensus* protocol which outputs a bit and a grade that serves as an indicator of confidence in the bit. Crusader agreement and graded agreement are well-known special cases of proxcensus with 3 and 5 grades respectively. If all honest parties input the same bit 0 (or 1), their output must be the minimal (or maximal) grade. All honest parties are guaranteed to have graded agreement

in the sense that they have adjacent integer outputs. After expanding to a large space of $2^\lambda + 1$ grades, the output bit is *extracted* by splitting the space in two using a random coin. If the obtained grade is strictly greater than the coin, the output is 1, otherwise it is 0. This means that if a party has the minimal (or maximal) grade, then they output 0 (or 1) regardless of the coin. Hence, the validity of BA reduces to the validity of proxcensus.

When we enter the extraction phase, a more general version of the attack on the FM approach in asynchrony described above also applies to asynchronous expand-and-extract. As we cannot wait to flip the coin until all honest parties are done expanding: each party must initiate the coin flip when they are done expanding and this leaks the value of the common coin to adversary. The output grades of proxcensus are guaranteed to be adjacent, but when the first honest party gives output, there are still potentially two candidates for the adjacent grade. This gives the adversary two shots at guessing the random coin instead of one, doubling the error probability. However, adding an extra round to expand to twice as many grades recovers this lost bit of security.

Finally, an additional round is used to reduce BA to by establishing a threshold signature on the input bit as suggested in [3].

1.1 Preliminaries

We provide protocols for n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ where $t < n/3$ can be adaptively Byzantine corrupted. We assume an asynchronous network where message delivery is handled by the adversary with no upper bound on the delay. Liveness properties hold under eventual delivery of the messages.

Threshold Signatures and Common Coin. These primitives are standard and we only give brief informal descriptions. For simplicity we treat them as unconditionally secure. We assume key shares of a threshold signature scheme for multiple different thresholds have been setup between the parties. We will be using thresholds $n - t$ and $t + 1$ and assume that (partial) signatures have a size of λ bits. We also assume the parties can run a common coin primitive Π_{CC} that allow the parties to flip a λ bit coin which is unpredictable to the adversary until the first honest party initiates the protocol. This can be instantiated from threshold signatures following [4].

Justifiers. We use justifiers as defined in [11]. These are ways to demonstrate some notion of validity of a protocol message by providing relevant context. A simple example is justifying the input bit to protocol using threshold signatures as suggested in [3].

Definition 1 (Justifier [11]). *For a message identifier* mid *we say that* J^{mid} *is a justifier if the following properties hold.* J^{mid} *is a predicate depending on the message* m *and the local state of a party. When we write pseudo-code then we write* $J^{\mathsf{mid}}(m)$ *to denote that the party* P *executing the code computes* J^{mid} *on* m *using its current state. In definitions and proofs we write* $J^{\mathsf{mid}}(m, \mathsf{P}, \tau)$ *to denote that we apply* J^{mid} *to* m *and the local state of* P *at time* τ*.*

Monotone: *If for an honest* P *and some time* τ *it holds that* $J^{\mathsf{mid}}(m, \mathsf{P}, \tau) = \top$ *then at all* $\tau' \geq \tau$ *it holds that* $J^{\mathsf{mid}}(m, \mathsf{P}, \tau') = \top$.

Propagating: *If for honest* P *and some point in time* τ *it holds that* $J^{\mathsf{mid}}(m, \mathsf{P}, \tau) = \top$*, then eventually the execution will reach a time* τ' *such that* $J^{\mathsf{mid}}(m, \mathsf{P}', \tau') = \top$ *for all honest parties* P'.

In our protocols all justifiers are explicit certificates based on threshold signatures and some auxiliary information send along with the message. We therefore omit the party and time from the justifiers. The following two definitions provide a framework to reason about properties of justified messages.

Definition 2 (Possible Justified Messages [11]). *Let* Π *be a protocol. When we say that an* ℓ*-ary predicate* P *holds* for all possible justified messages *we mean: Run the protocol* Π *under attack by the adversary. At some point the adversary may output a sequence of triples* $(\mathsf{P}^1, \mathsf{mid}^1, m^1), \ldots, (\mathsf{P}^\ell, \mathsf{mid}^\ell, m^\ell)$*. We say that the adversary wins if the message identifiers* $\mathsf{mid}^1, \ldots, \mathsf{mid}^\ell$ *identify messages of* Π*,* $\mathsf{P}^1, \ldots, \mathsf{P}^\ell$ *are honest (but not necessarily distinct) parties, for* $j = 1, \ldots, \ell$ *it holds that* $J^{\mathsf{mid}_j}(m^j) = \top$ *at* P^j*, and* $P(m^1, \ldots, m^\ell) = \bot$*. Otherwise the adversary looses the game. Any PPT adversary should win with negligible probability.*

Definition 3 (Possible Justified Outputs [11]). *Let* Π *be a protocol with output justifier* J*. When we say that an* ℓ*-ary predicate* P *holds* for all possible justified outputs *we mean: Let* Π' *be the protocol* Π *with only change being that each party on getting output, sends their output to all parties if this was not already done. Run the protocol* Π' *under attack by the adversary. At some point the adversary may output a sequence of triples* $(\mathsf{P}^1, \mathsf{mid}^1, m^1), \ldots, (\mathsf{P}^\ell, \mathsf{mid}^\ell, m^\ell)$*. We say that the adversary wins if the* $\mathsf{mid}^1, \ldots, \mathsf{mid}^\ell$ *are identified with outputs of* Π*,* $\mathsf{P}^1, \ldots, \mathsf{P}^\ell$ *are honest (but not necessarily distinct) parties, for* $j = 1, \ldots, \ell$ *it holds that* $J^{\mathsf{mid}_j}(m^j) = \top$ *at* P^j*, and* $P(m^1, \ldots, m^\ell) = \bot$*. Otherwise the adversary looses the game. Any PPT adversary should win with negligible probability.*

The protocol in Fig. 1 corresponds to the procedure for justifying inputs in [3]. The parties multicast their input bits and initially try to collect signature shares on some input bit from $t + 1$ parties, implying that it is the input of an honest party. At the same time (assuming $n > 3t$) it is guaranteed that parties eventually receive $t + 1$ shares for one of the two inputs, since the bit input by the majority of the honest parties will account for at least $t + 1$ shares. Using the definition of justifiers from [11] we can say that the Π_{IVG} in Fig. 1 has the liveness property that if all honest parties start running the protocol, then eventually they receive a justified bit as output. It also has the safety property that any *possible justified output* satisfying $\Pi_{\text{IVG}}.J_{\text{OUT}}$ was the input of an honest party.

2 Validated Proxcensus

In this section we define validated proxcensus, which is to the definition of proxcensus in [9] what [3] is to BA. To ease the notation in the protocol and proofs

Input validation gadget $\Pi_{\mathbf{IVG}}$.

- On input b_i, P_i multicasts b_i with a signature and a partial signature with threshold $t+1$.
- On receiving b for some bit b from $t+1$ distinct parties: P_i combines the shares of the $t+1$ threshold signature scheme and outputs b justified by the threshold signature.

Fig. 1. An input validation gadget reducing BA to VBA.

we define the output over the integers, such that $VProx-(G)$ has outputs in $\{0,\ldots,G-1\}$ rather than a bit (or '?') with a grade up to $\lfloor G/2 \rfloor$.

We will follow the approach of reducing the problem of $VProx-(2s-1)$ to $VProx-(s)$ using one round of communication.

Definition 4 (Validated Proxcensus). *Let $\Pi_{VPROX-(G)}(J_{IN})$ be a protocol for n parties, parameterized by an input justifier J_{IN}, and outputting $y \in \{0,\ldots,G-1\}$ satisfying an output justifier J_{OUT}. We say that $\Pi_{VPROX-(G)}(J_{IN})$ solves VProx-(G) if the following holds:*

Liveness *If every honest party P_i have justified input $x \in \{0,1\}$ where $J_{IN}(x) = \top$, then eventually every honest party P_j will have justified output $y \in \{0,\ldots,G-1\}$ where $J_{OUT}(y) = \top$.*

Justified Graded Agreement *For all possible justified outputs y and y': $|y-y'| \leq 1$.*

Justified Validity *If b is the only possible J_{IN} justified bit, then $y = b \cdot (G-1)$ for all possible justified outputs y.*

Remark 1. It is easy to map from an output $y \in [G]$ to the (b,g) representation used in [9]. In our case G is always odd, so we can define a middle grade $G' = \lfloor G/2 \rfloor$ and map y to (b,g) where $g = |G'-y|$ and $b = \begin{cases} ? & \text{if } y = G' \\ 0 & \text{if } y < G' \\ 1 & \text{if } y > G' \end{cases}$.

To motivate the definition and the use of justifiers, let us first consider the simplest (non-trivial) version of proxcensus: Crusader Agreement. Parties have input in $\{0,1\}$ and outputs in $\{0,1,2\}$. If we use the logic from Fitzi et al. in the asynchronous setting we run into problems with validity, liveness or agreement (unless we assume fewer corruptions or use more than one round). First, a party should not output $2b$ unless it has seen $n-t$ votes for b. Otherwise graded agreement is easily broken. The validity property says that if all honest parties have input b, then the only valid output is $2b$. In order to make sure that you pick a valid output you can only "trust" your own input, or the other input if it is seen from at least $t+1$ parties. But what do you do if you had input b, only see $n-t$ votes, and the majority but not all of these votes are on b? You do not have enough information to know that $1-b$ is an honest input and thus that 1 is a valid output, but neither do you have the $n-t$ votes for b that would allow

outputting $2b$. As the protocol is asynchronous, there is no way to wait for more than $n - t$ votes. While this is far from a formal lower bound, it illustrates the problem which occurs because we cannot wait for all honest votes as you could in the synchronous setting. It can be solved by assuming $n > 4t$, or by using two rounds in each step of the recursion.[1]

In order to solve this problem, let us define a *validated* flavour of Crusader Agreement which is to Crusader Agreement what is to BA. Namely, where inputs are checked by a predicate and the validity property only holds with respect to the predicate. We define Justified Validity to say that if only the input b is justifiable, then $2b$ should be the only justifiable output bit. With this definition: if you see a justifier for both possible input bits, then you can output bottom without violating Justified Validity. You can also justify it by forwarding the 2 justifiers. To ensure agreement you can now simply wait until you see $n-t$ votes for the same bit b in order to output $2b$. Finally, for Liveness observe that you will eventually see $n-t$ votes from honest parties. Either these $n-t$ votes are on the same bit, or you saw two that were different votes. In either case you obtain an output.

Let us generalize this idea to solve validated proxcensus. We now reinterpret the consensus rules as doubling an input that is seen n-t times, or taking the double of the average of two different inputs. Note that after giving output: since all justified outputs are on at most two different different adjacent integers, we are in an abstract sense in the same situation as when we started with 0 and 1 being those two integers. If we apply this new interpretation of the consensus rules to those integers and use the invariant that all justifiable outputs of an iteration are adjacent integers, we can keep threading the justified outputs of the each instance of validated crusader agreement into the next one. The maximum grade doubles every round. To see why justified validity is maintained: there is only something to show if the only justified input is ß. And in that case each round just doubles the unique justified input and the only justified output after r rounds will be $2^r \cdot b$.

We describe the general protocol in detail in Fig. 2 as reducing VProx-$(2s-1)$ to VProx-(s). As a base case for the recursion, define $\Pi_{\text{VPROX}-(2)}(J_{\text{IN}})$ to output its input with $J_{\text{OUT}} = J_{\text{IN}}$. Rephrasing the above: the main insight is that *all possible justified* outputs of VProx-(s) are on at most 2 adjacent grades, so we can double the grade if we only receive the same grade from $n - t$ parties, or double the average of 2 justified grades.

We show that the protocol in Fig. 2 satisfies VProx-$(2^i + 1)$ as defined in Definition 4 for any nonnegative integer i, using i rounds of communication. For the base case $i = 0$ define $\Pi_{\text{VPROX}-(2)}(J_{\text{IN}})$ to be the zero round protocol that just returns the input with output justifier $J_{\text{OUT}} = J_{\text{IN}}$ trivially satisfying Definition 4. We show the induction step:

[1] After we introduce the proxcensus protocol it should be clear how to combine it with the protocol in Fig. 1 to obtain a recursive solution with to rounds in each step. But also that the "extra" round is only needed in the beginning of the protocol.

Reduction from $\Pi_{\mathbf{VProx}-(2s-1)}$ to $\Pi_{\mathbf{VProx}-(s)}$ for $s \geq 2$.

- On input x_i where $J_{\text{IN}}(x_i) = \top$, P_i runs $\Pi_{\text{VPROX}-(s)}(J_{\text{IN}})$ with input x_i.
- On output z_i from $\Pi_{\text{VPROX}-(s)}(J_{\text{IN}})$, P_i multicasts (PROPOSAL, z_i) with a signature and a partial signature with threshold $n - t$.
- On receiving justified (PROPOSAL, z) and (PROPOSAL, $z + 1$) P_i lets $y_i = 2z + 1$ and terminates with output y_i justified by the justifiers for z and $z + 1$.
- On receiving justified (PROPOSAL, z) from $n - t$ distinct parties, P_i lets $y_i = 2z$ and terminates with output y_i justified by a $n - t$ threshold signature on z.

Fig. 2. A recursive description of the validated proxcensus protocol.

Lemma 1. *If $\Pi_{VPROX-(s)}(J_{IN})$ satisfies Definition 4, then $\Pi_{VPROX-(2s-1)}(J_{IN})$ satisfies Definition 4*

Proof. For liveness we observe that $n - t$ honest parties $\Pi_{\text{VPROX}-(s)}(J_{\text{IN}}).J_{\text{OUT}}$ justified values z_i which by Justified Graded Agreement are either all identical or split between two adjacent integers, so when these are propagated every party has a set of $n - t$ values that allow defining y_i through one of the two cases. Justified Validity follows from Justified Validity of $\Pi_{\text{VPROX}-(s)}(J_{\text{IN}})$, as the input justifier is shared. So since all justified z are identical, every party gets the same y. In particular, if the only justified input to the inner protocol is b then the only $\Pi_{\text{VPROX}-(s)}(J_{\text{IN}}).J_{\text{OUT}}$-justified output is $z = b \cdot (s - 1)$, thus the only $\Pi_{\text{VPROX}-(2s-1)}(J_{\text{IN}}).J_{\text{OUT}}$-justified output is $2b \cdot (s - 1) = b \cdot (2s - 2)$. For Justified Graded Agreement we again rely on Justified Graded Agreement of the $\Pi_{\text{VPROX}-(s)}(J_{\text{IN}})J_{\text{OUT}}$-justified z values to say that and for any justified z_i and z_j we have $|z_i - z_j| \leq 1$. We only need to argue that there cannot be $n - t$ parties who send some z_i and $n - t$ parties who sent $z_j = z_i - 1$. But such two sets would overlap on at least one honest party as $n > 3t$. Thus, by definition of step 3 all justified outputs y_i and y_j satisfy $|z_i - z_j| \leq 1$. □

In Appendix A we show that the justifiers remain constant size which allows our asynchronous validated proxcensus to match the complexity of the synchronous proxcensus for $n > 3t$ parties presented in [9].

Corollary 1. *For any $r \geq 0$, $\Pi_{VPROX-(2^r+1)}(J_{IN})$ solves VProx-($2^r + 1$) with r rounds of communication and $O(rn^2(\lambda + |J_{IN}|))$ bits of communication, where $|J_{IN}|$ is the size of the input justifier. When using Π_{IVG} to establish J_{IN}, it solves Prox-(2^r+1) as defined in [9] in $r+1$ rounds with $O(rn^2\lambda)$ bits of communication.*

3 Validated BA

We give a brief description of the extraction phase, which largely follows [9]. The expansion step requires an extra round of communication to expand to $2^{\lambda+1} + 1$ grades and make up for the error probability being doubled in asynchrony. (In

contrast to the $2^\lambda + 1$ that suffice in the synchronous case.) As the proxcensus is validated, the combined expand-and-extract procedure only yields a validated BA protocol. A full BA protocol is given in Section 4. We first define validated BA.

Definition 5 (Validated BA). *Let $\Pi_{VBA}(J_{IN})$ be a protocol for n parties parameterized by an input justifier J_{IN} outputting $y \in \{0,1\}$ satisfying an output justifier J_{OUT}. We say that $\Pi_{VBA}(J_{IN})$ is a secure protocol if the following properties hold:*

Liveness *If every honest party P_i has justified input $x_i \in \{0,1\}$ where $J_{IN}(x_i) = \top$, then eventually every honest party P_j will have justified output $y \in \{0,1\}$ where $J_{OUT}(y) = \top$.*

Justified Agreement *For all possible justified outputs y and y': $y = y'$.*

Justified Validity *If $J_{OUT}(y) = \top$, then $J_{IN}(y) = \top$.*

The protocol Π_{VBA} in Fig. 3 satisfies Definition 5 except with probability $2^{-\lambda}$. As mentioned the parties first expand to $2^{\lambda+1} + 1$ grades. Then Π_{CC} is run to obtain output c and the grades are compared with $2c$. This mitigates the security loss caused by the adversary being free to choose between more than two grades at the time the coin is flipped. The adversary can still learn c when the first honest party P_i gets their grade g_i and then decide to give some P_j grade $g_j \in \{g_i - 1, g_i, g_i + 1\}$ based on the value of c. But note that since $2c$ is even, agreement can only be broken if there are justified grades g and $g' = g + 1$ where g is even. So even if the adversary has the ability to to choose between $g_i - 1$ and $g_i + 1$ becoming justified grades after learning the value of c. At most one of the two grades can result in conflicting bit decisions, so the adversary has no chance of breaking agreement beyond guessing the exact value of c before any honest party initiates Π_{CC}.

Validated BA protocol Π_{VBA}

- On input b_i, P_i initiates $\Pi_{\text{VPROX}-(2^{\lambda+1}+1)}$ with input b_i.
- On output g_i from $\Pi_{\text{VPROX}-(2^{\lambda+1}+1)}$, P_i initiates Π_{CC}.
- On output c_i from Π_{CC}, P_i lets $d_i = 1$ if $g_i > 2c_i$.

Fig. 3. A validated BA using expand-and-extract.

Theorem 1. *Π_{VBA} (Fig. 3) is a secure as defined in Definition 5 except with probability $2^{-\lambda}$. It uses $\lambda + 2$ rounds of communication and $O(\lambda n^2(\lambda + |J_{IN}|))$ bits of communication, where $|J_{IN}|$ is the size of the input justifier.*

Proof. Since we expand to $2^{\lambda+1} + 1$ grades and double the value of the coin: even if the adversary can choose between 3 different grades when the coin is leaked, the adversary needs to guess the value of the coin to split agreement. Justified validity reduces to justified validity of validated proxcensus: If b is the only justified input, then the only justified possible justified grade is $b \cdot (G - 1)$ and the only possible justified output is b.

4 Binary Agreement

We finally solve BA using the recipe from [3] where a justifier is formed using a threshold signature before running . This resulting BA has a slightly stronger security definition than usual: the output is justified and all possible justified outputs are identical.

Binary Agreement protocol $\Pi_{\mathbf{BA}}$.

- On input b_i, P_i initiates Π_{IVG} with input b_i.
- On output x_i from Π_{IVG}, P_i initiates $\Pi_{\text{VBA}}(\Pi_{\text{IVG}}.J_{\text{OUT}})$ with input x_i.
- On output y_i from d_i $\Pi_{\text{VBA}}(\Pi_{\text{IVG}}.J_{\text{OUT}})$, P_i outputs d_i.

Fig. 4. A Binary Agreement protocol.

Definition 6 (BA). *Let Π_{BA} be a protocol for n parties outputting $y \in \{0,1\}$ satisfying an output justifier J_{OUT}. We say that Π_{BA} is a secure BA protocol if the following properties hold:*

Liveness *If every honest party P_i has input $x_i \in \{0,1\}$, then eventually every honest party P_j will have justified output $y_i \in \{0,1\}$ where $J_{OUT}(y_i) = \top$.*
Justified Agreement *For all possible justified outputs y and y': $y = y'$.*
Validity *If $J_{OUT}(y) = \top$, then some honest party gave input y.*

We give a protocol Π_{BA} in Fig. 4 which implements BA except with probability $2^{-\lambda}$. The security follows as a corollary from Theorem 1.

Corollary 2. *Π_{BA} given in Fig. 4 implements a secure BA as defined in Definition 6 except with probability $2^{-\lambda}$. It uses $\lambda + 3$ rounds of communication and $O((n\lambda)^2)$ bits of communication.*

4.1 Related work

Expand-and-extract along with the notion of proxcensus were introduced in [9]. Proxcensus is in turn an adaption of the proxcast definition used in [5]. Justifiers were introduced in [6], but we use the definitions from [11]. A very similar notion of transferable justifiers that also considers a version of adversarially crafted justified outputs exist in [13], however we need to apply it to properties of messages rather than just outputs, which matches the security game in [11]. The logic used to get validated proxcensus draws on the partially synchronous BA in [12], although the network model and definitions of justifiers are quite different. The result of [9] has later been improved in [10], but the techniques do not appear to be compatible with asynchrony. There has not been much recent progress on the concrete round efficiency of Monte Carlo style BA in the asynchronous setting. However, the recent work of Erbes and Wattenhofer [7] provides a 2^{λ} graded consensus protocol with $6(\lambda + 1)$ rounds and suggests that [1] and [2] can be combined into a 2^{λ} graded consensus using $3(\lambda+1)$ rounds. We improve on this by a factor of 3 (see Corollary 1).

References

1. Attiya, H., Censor-Hillel, K.: Lower bounds for randomized consensus under a weak adversary. SIAM J. Comput. **39**(8), 3885–3904 (2010)
2. Bandarupalli, A., Bhat, A., Bagchi, S., Kate, A., Liu-Zhang, C., Reiter, M.K.: Delphi: Efficient asynchronous approximate agreement for distributed oracles. In: DSN. pp. 456–469. IEEE (2024)
3. Cachin, C., Kursawe, K., Petzold, F., Shoup, V.: Secure and efficient asynchronous broadcast protocols. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 524–541. Springer, Berlin, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2001). https://doi.org/10.1007/3-540-44647-8_31
4. Cachin, C., Kursawe, K., Shoup, V.: Random oracles in Constantinople: Practical asynchronous byzantine agreement using cryptography. Journal of Cryptology **18**(3), 219–246 (Jul 2005). https://doi.org/10.1007/s00145-005-0318-0
5. Considine, J., Fitzi, M., Franklin, M.K., Levin, L.A., Maurer, U.M., Metcalf, D.: Byzantine agreement given partial broadcast. J. Cryptol. **18**(3), 191–217 (2005). https://doi.org/10.1007/S00145-005-0308-X, https://doi.org/10.1007/s00145-005-0308-x
6. Dinsdale-Young, T., Magri, B., Matt, C., Nielsen, J.B., Tschudi, D.: Afgjort: A partially synchronous finality layer for blockchains. In: Galdi, C., Kolesnikov, V. (eds.) SCN 20. LNCS, vol. 12238, pp. 24–44. Springer, Cham, Switzerland, Amalfi, Italy (Sep 14–16, 2020). https://doi.org/10.1007/978-3-030-57990-6_2
7. Erbes, M.M., Wattenhofer, R.: Asynchronous approximate agreement with quadratic communication (2024)
8. Feldman, P., Micali, S.: An optimal probabilistic protocol for synchronous byzantine agreement. SIAM J. Comput. **26**(4), 873–933 (1997)
9. Fitzi, M., Liu-Zhang, C.D., Loss, J.: A new way to achieve round-efficient byzantine agreement. In: Miller, A., Censor-Hillel, K., Korhonen, J.H. (eds.) 40th ACM PODC. pp. 355–362. ACM, Virtual Event, Italy (Jul 26–30, 2021). https://doi.org/10.1145/3465084.3467907
10. Ghinea, D., Goyal, V., Liu-Zhang, C.D.: Round-optimal byzantine agreement. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part I. LNCS, vol. 13275, pp. 96–119. Springer, Cham, Switzerland, Trondheim, Norway (May 30 – Jun 3, 2022). https://doi.org/10.1007/978-3-031-06944-4_4
11. Kamp, S.H., Nielsen, J.B.: Byzantine agreement decomposed: Honest majority asynchronous total-order broadcast from reliable broadcast. IACR Cryptol. ePrint Arch. p. 1738 (2023)
12. Kamp, S.H., Nielsen, J.B., Thomsen, S.E., Tschudi, D.: Enig: Player replaceable finality layers with optimal validity. Cryptology ePrint Archive, Report 2022/201 (2022), https://eprint.iacr.org/2022/201
13. Loss, J., Nielsen, J.B.: Early stopping for any number of corruptions. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part III. LNCS, vol. 14653, pp. 457–488. Springer, Cham, Switzerland, Zurich, Switzerland (May 26–30, 2024). https://doi.org/10.1007/978-3-031-58734-4_16

A Proxcensus with constant sized justifiers of odd grades

The proof of Lemma 1 completes the induction proof as far as correctness and round complexity goes. But notice that while each party only sends a single

message in each round, the size of the odd justifiers a priori keep growing, because they depend recursively on justifiers from the previous round. So, the size of the output justifier of an odd grade in round i is $O(i\lambda + |J_{\text{IN}}|)$ bits, where $|J_{\text{IN}}|$ is the size of the input justifier. Meanwhile the justifier of an even grade is compressing in the sense that it comes with a threshold signature which was produced by a set of parties including at least one honest party, and thus is of size $O(\lambda)$ bits as it does not need to include justifiers from previous rounds. Let $\Pi_i = \Pi_{\text{VProx}-(2^i+1)}$ for $i \geq 0$. For Π_0 the output justifier is the input justifier of size $O(|J_{\text{IN}}|)$. To optimize the size of odd output justifiers to $O(\lambda + |J_{\text{IN}}|)$ for $i > 0$, observe that the justifier of an odd grade is based on a justified even and odd grade from the previous round. Consider an odd output y of Π_i, which was calculated as $2z - 1$ based on justified values z and $z-1$. We can think of this as $y = 2\frac{|z'+z''|}{2}$ for some justified values z' and $z'' = z' \pm 1$. Assume W.L.O.G. that z' is the even of the two grades, then it was justified by an $n-t$ threshold signature on $z'/2$ while z'' was justified by the same threshold signature justifier for $z'/2$ in addition to the justifier for the odd grade $z'/2 \pm 1$. Since the threshold signature justifying z' has contributions from honest parties who saw a justifier for $z'/2$, it guarantees the existence of the justifier for $z'/2$. This means that – in the context of justifying y – the justification of z'' only needs to justify its $z'/2 \pm 1$ component. If $z'/2 \pm 1$ is even, it is justified by a threshold signature on $(z'/2 \pm 1)/2$ (or is 0 and justified by J_{IN}). Otherwise, $z'/2 \pm 1$ is odd and we apply the above step until we hit another threshold signature or an input justified bit. In summary, the output for any grade at any level can be justified using $O(\lambda + |J_{\text{IN}}|)$ bits.